CP1406 Week 7 Practical – HTML Tables and Responsive Content & Sign Up to SiteGround

A table presents related information in rows and columns, and is especially useful when comparing types of data or listing topics and details. Tables can use column headings or row headings to identify categories or topics. Tables are also helpful when you need to provide a lot of content in a compact format. For this week's practical, you will create a simple responsive table.

For today's practical, you need to submit:

- A single zip file of the **Week 7 Prac** folder. The folder includes:
  - one (1) html file (cms.html) you have created for this week prac
  - a CSS file (tablestyle.css)

Part 1 – Creating Tables with HTML

Table Building Blocks

Tables are created by nesting a variety of elements between table tags. Tables are organised into rows, not columns, by the table row (tr) element. Each table row is made up of one or more table data (td) entries (or table header (th)). Columns are formed automatically when table data elements from each subsequent table row automatically line up in vertical columns.

Those three main elements, table, tr, and td, are the basic building blocks of HTML tables. Here's an example of how you can use just those three elements to create a simple table:

```
<!doctype html>
<html>
<head>
        <meta charset="utf-8">
        <title>Simple Table</title>
</head>

<body>
        <table>
                <tr>
                        <td>Mood</td>
                        <td>Colour</td>
                        <td>Weather</td>
                </tr>
                <tr>
                        <td>Happy</td>
                        <td>Yellow</td>
                        <td>Sunny</td>
                </tr>
```

```
            <tr>
                <td>Sleepy</td>
                <td>Grey</td>
                <td>Cloudy</td>
            </tr>
        </table>
</body>
</html>
```

- Create a folder called "Week 7 Prac".
- Type or copy and paste the code from above to create a simple 3 row, 3 column table.
- Save the file inside the "Week 7 Prac" as "simpletable.html".
- Open simpletable.html in a web browser and see the page.

Using these simple building blocks, you can build many data tables. However, sometimes you need a more complex structure such as that listed below (Figure 1):

CMS	Usage	Change since 1 January 2022	Market Share	Change since 1 January 2022
WordPress	43.30%	0.10%	65.30%	0.10%
Shopify	4.50%	0.10%	6.70%	0.10%
Wix	2.00%	0.10%	2.90%	No Change
Squarespace	1.80%	No Change	2.70%	No Change
Joomla	1.70%	No Change	2.50%	-0.10%
Total	53.30%		80.10%	
The data in this table is provided courtesy of W3Techs and was captured in February 2022.				

Figure 1

The order of the table elements should be:

1	<table>	Start table
2	<caption>	Do caption
3	<thead> <tr><th>first column</th> <th>second column</th> </tr> </thead>	Do table header • header tag • row tag • use th for each column heading
4	<tbody> <tr> <td> Data content </td> <td>One cell for each element</td> </tr> </tbody>	Do table body • table body tag • start row • start cells • end row

		• end table body
5	<tfoot> <tr> <td> Data content </td> <td>One cell for each element </td> </tr> </tfoot>	Do table footer • footer tag • start row • start cells • finish off row • end footer tag
6	</table>	End the table

Table 1 The Order of Table Elements

You will now recreate the table shown in Figure 1.

- Open a new file, save this file as "**cms.html**" inside the "**Week 7 Prac**" folder.
- Captions help users to find a table and understand what it's about and decide if they want to read it.
  Create a table caption so that screen readers can explain the data.

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>CMS</title>
</head>
<body>

    <table>
        <caption>Most Popular Content Management Systems</caption>
    </table>

</body>
</html>
```

- To add a table header, you use the code:

```
<table>
    <caption>Most Popular Content Management Systems</caption>
        <thead>
            <tr>
                <th>CMS</th>
                <th>Usage</th>
                <th>Change Since Jan 1</th>
                <th>Market Share</th>
```

```
                <th>Change Since Jan 1</th>
            </tr>
        </thead>
    </table>
```

- Add content of the table above (see Figure 1) using HTML code. See **Table 1 - <tbody>** element.
- Now that you have all the table content, update your table by adding the table footer. See **Table 1 - <tfoot>** element.

You also should add an attribution to where the data came from and a bit of explanation about the data. You will add this to your footer information by spanning the columns.

- Add an extra row in your footer with the following code:

```
<tr>
    <td colspan="5">The data in this table is provided courtesy of <a
    href="http://w3techs.com" target="_blank">W3Techs</a> and was captured
    in February 2022.</td>
</tr>
```

Since our table has 4 cells in one column with "No Change" listed, it would be good to combine them into one cell. In cases where we want to span a few rows (rather than columns) we use rowspan and designate how many rows we want to span.

- Go to the **Squarespace** row, and for the cell with “No Change” content use this code:

```
<td rowspan = "2">No Change</td>
```

- Repeat above step with the **Wix** row, last column with the “No Change” content.
- For the Joomla and Squarespace row, delete the cell (<td></td> with the “No Change” content.
- You table should now look similar to Figure 2:

Note: This view has a table border set to 1, just so we can see where the cells begin and end. You will work on styling this table in Part 2 – Styling Tables with CSS.

Most Popular Content Management Systems

CMS	Usage	change since 1 January 2022	Market Share	change since 1 January 2022
WordPress	43.3%	+0.1%	65.3%	+0.1%
Shopify	4.5%	+0.1%	6.7%	+0.1%
Wix	2.0%	+0.1%	2.9%	No Change
Squarespace	1.8%	No Change	2.7%	
Joomla	1.7%		2.5%	-0.1%
Total	53.30%		80.1%	
The data in this table is provided courtesy of W3Techs and was captured in February 2022.				

Figure 2

Part 2 – Styling Tables with CSS

The current table that you have just created is still rather bare bones. There are several things you could do to improve it (you may think of others!):

- ✓ The caption needs to increase in size so that it's more prominent.
- ✓ Table heading elements could be larger to make them stand out more
- ✓ Increase the size of the first column of table data elements
- ✓ Footer text should be small, like a footnote
- ✓ Text should be centred in all the cells
- ✓ Table should be responsive to different viewport sizes

As you may recall, according to the HTML 5 standard, all styling and positioning of HTML elements should occur in the CSS. This holds true for tables as well.

- Create a stylesheet called tablestyles.css and link your page to it.
- On the stylesheet, let's set the table caption to font-size of 1.5em and add some padding.

```
/*Targets the caption text*/
table caption {
    font-size: 1.5em;
    padding: 10px 5px;
}
```

- Next, look at the CSS code below and add it to your stylesheet.
  Both rules use a special selector called nth-child.

```
/*Targets the headings and first column*/
th, td:nth-child(1) {
     font-size: 1.1em;
```

```
    font-weight: 500;
}

/*Targets footer explanation and attribution*/
tfoot tr:nth-child(1n+2) {
    font-size: .9em;
    font-weight: normal;
    text-align: left;
}
```

The nth-child selector is used to pinpoint elements based on their position in relation to their parent element. There are some useful values we can use for the nth-child selector:

- Numbers: identify a specific child element of a parent.
  For instance, in the first section, we can use td:nth-child (1) to get the first child element of the <td> tag. By targeting the <td> element we will affect ALL <td> elements in the first column.

- Words: like odd or even can be used to target every other child element
  This is often used to create striping on rows to make them easier to distinguish from each other.

- Formulas using format an+b: targets a child element that is a result of the formula
  For example our css code uses 1n+2. This will target every child element which exists as a possible solution to the equation. This includes every child element other than the first.

  Replace n with "0" and we get 1*0+2=2
  Replace n with "1" and we get 1*1+2=3
  Replace n with "2" and we get 1*2+2=4

- View your HTML file in a web browser now. You should see the caption increased in size, and the bottom line of your footer is smaller than the previous line.
- Next, add CSS to centre the contents each table cell and to add 5px of padding around it.

```
td {
    text-align: center;
    padding: 5px;
}
```

- Try your page again and make sure the CSS has taken effect.
  Note: prior to the invention of CSS, any styling or font changes in tables had to be made in each. and. every. cell. of. the. table. Very tedious. CSS is MUCH FASTER.
- Next, we will do a bit more editing so that we can separate out our rows of data.
  Add the following CSS to your styles file and now run your HTML page again.

As you do this, make sure you are ***understanding*** what the code actually does.

```
/*Remove the borders from all table elements*/
table, th, td {
    border: none;
}

/*Apply a background colour to the table heading*/
thead {
    background-color: #a7a37e;
}


/*Apply a background colour to every other row in the table body*/

tbody tr:nth-child(even) {
    background-color: #efecca;
}

/*Apply a background colour to every other row in the table footer*/

tfoot tr:nth-child(odd) {
    background-color: #efecca;
}
```

The first two sections are quite straight-forward, remove the borders and apply a background colour to the table head.

The second half of the additions use the nth-child elements using odd and even. Try these colours out first and then edit your own CSS file to use some different ones that still look nice together.

- The table doesn't look too bad at this point if we view it on a laptop, but it's nearly unreadable on a smaller device. We need to make our page responsive so that it can handle being viewed in other situations.
- The first thing we need to do is create a rule for viewports (screens) that are 780px or smaller.
- Add this code to your CSS file:

```
@media only screen and (max-width: 780px) {
/*CSS styles will be added here*/
 }
```

- Below the CSS comment in the media section, add CSS to change the table elements into block level items so they can be split apart and positioned where we wish.

```
table, thead, tbody, tfoot, tr, th, td {
    display: block;
}
```

You will be adding new labels for the td elements using CSS, but you should keep the existing headers available for screen readers. The way to do this is to position the header elements offscreen far enough that they don't show up on the smaller screen, but close enough for assistive technologies to find them.
You also need to move the Totals row offscreen as it's not critical to understanding the contents of the table.

- Add the following CSS code under your last @media entry:

```
thead tr, tfoot tr:nth-of-type(1){
    position: absolute;
    top: -9999px;
    left: -9999px;
}
```

You have now uncoupled your row elements so they will appear stacked on top of one another.
Now you will apply a 50% padding, so they appear in the middle of the screen.
You use **position: relative** because when you use **position: absolute** with the data labels, they will display relative to the data cell.

- Add the following CSS code under your last @media entry:

```
tfoot tr td {
    padding-left: 0%;
    text-align: left;
 }
```

Due to the changes you made in the styles for the last step, you need to write some specific styles for the data in the footer. The two rules will make the text in the footer be left-aligned and take up the width of the table.

- Although your actual headings are hidden just off-screen, you can use CSS to position new data labels next to your table body <td> elements with these rules:

```
td:before {
    top: 5px;
    left: 5px;
    white-space: nowrap;
    text-align: left;
}
```

- Finally, we apply some rules to put print labels to the left of the data displayed. Add this code to your CSS file:

```
tbody td:nth-child(1):before {
      content: "CMS";
}

tbody td:nth-child(2):before {
      content: "Usage*";
}

tbody td:nth-child(3):before {
     content: "+/- since Jan";
}

tbody td:nth-child(4):before {
     content: "Market Share*";
}

tbody td:nth-child(5):before {
     content: "+/- since Jan";
}
```

- Try resizing the browser to see the effects of these rules.

Part 2 – Get an account on SiteGround for your website

We will create an account on SiteGround to obtain web hosting space from SiteGround (free for 3 months). You will upload your assignment to the SiteGround web server later in the semester.

- Go to https://www.siteground.com/studentsprogram. Follow the instructions to create a user account. You must use your JCU student email as your email address.
- Check your JCU email for a confirmation email from SiteGround. It will contain important information you need to login to your new web space.
- Login to your web space using your SiteGround login. Make a note of your domain name. It will be something like: yourname.sgedu.site.

Deliverables:

- A single zip file of the **Week 7 Prac** folder. The folder includes:
    - one (1) html file (cms.html) you have created for this week prac
    - a CSS file (tablestyle.css)